IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of

GUSTAVSON, et al.

Serial No.:	10/671,887	Group Art Unit:	2193
Filed:	September 29, 2003	Examiner:	Ngo, Chuong D.

For: METHOD AND STRUCTURE FOR PRODUCING HIGH PERFOMANCE LINEAR ALGEBRA ROUTINES USING COMPOSITE BLOCKING BASED ON L1 CACHE SIZE

Commissioner of Patents
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

Sir:

Appellants respectfully appeal the rejection of claims 1, 2, and 4-27 in the Office Action mailed on October 31, 2007. A Notice of Appeal was timely filed on January 30, 2008.

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, assignee of 100% interest of the above-referenced patent application.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1, 2, and 4-27 stand rejected under 35 U.S.C. § 101 as allegedly directed to non-statutory subject matter. Claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27 stand rejected under 35 U.S.C. § 112, second paragraph, as allegedly indefinite. Claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27 stand rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,099,447 to Myszewski, further in view of US Patent Publication 2003/0088600 to Lao et al.

All rejections are being appealed.

IV. STATUS OF AMENDMENTS

An Amendment Under 37 CFR §1.116 was filed on December 31, 2007. In the Advisory Action mailed on January 29, 2008, the Examiner indicated that the arguments in the Amendment Under 37 CFR §1.116 were not persuasive and that the rejections of record were maintained for all claims.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Bases in the specification for the independent claims:

1. (Rejected) A method of increasing at least one of efficiency and speed in executing a matrix subroutine on a computer, said method comprising:

storing data contiguously for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache (Figure 3, note that 2NB is twice the dimension of cache) and a second dimension of said block being smaller than a corresponding second dimension of said cache (Figure 3, note that NB/2 is ½ the cache dimension), such that said block fits into a working space of said cache (Figure 3, note that NB/2 x 2NB fits into the cache working space; line 18 of page 11 through line 2 of page 12).

10. (Rejected) An apparatus, comprising:

a processor (111 of Figures 1 & 2), for processing a matrix subroutine;

a cache (201 of Figure 2) associated with said processor; and

a memory (121 of Figure 1), wherein said memory stores data for memory calls of said matrix subroutine as contiguous data in an increment block size that is based on a dimension of said cache and loads said blocks of data into said cache for said matrix subroutine processing, wherein a dimension of said increment block size is larger than any dimension of a working area of said cache used for processing said matrix subroutine (Figure 3, note that 2NB is twice the dimension of cache).

15. (Rejected) A computer program product (150 of Figure 1) for use with a computer, said computer program product comprising a machine-readable medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus, said instructions including a method of storing data for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache (Figure 3, note that 2NB is twice the dimension of cache) and a second dimension of said block being smaller than a corresponding second dimension of said cache (Figure 3, note that NB/2 is ½ the cache dimension), such that said block fits into a working space of said cache (Figure 3, note that NB/2 x 2NB fits into the cache working space; line 18 of page 11 through line 2 of page 12).

19. (Rejected) A method of solving a problem using linear algebra, said method comprising at least one of:

initiating a computerized method of performing one or more matrix subroutines (lines 8-11 of page 19), wherein said computerized method comprises storing data for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache (Figure 3, note that 2NB is twice the dimension of cache) and a second dimension of said block being smaller than a corresponding second dimension of said cache (Figure 3, note that NB/2 is ½ the cache dimension), such that said block fits into a working space of said cache (Figure 3, note that NB/2 x 2NB fits into the cache working space; line 18 of page 11 through line 2 of page 12);

transmitting a report from said computerized method via at least one of an internet interconnection and a hard copy (lines 14-15 of page 19); and

receiving a report from said computerized method (lines 16-19 of page 19).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellant presents the following three grounds for review by the Board of Patent Appeals and Interferences:

GROUND 1: <u>The rejection for claims 1, 2, and 4-27, based on 35 U.S.C. § 101</u>.

GROUND 2: <u>The rejection for claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27, based on 35 U.S.C. § 112, second paragraph</u>.

GROUND 3: <u>The rejection for claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27, based on 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,099,447 to Myszewski, further in view of US Patent Publication 2003/0088600 to Lao et al.</u>

VII. ARGUMENTS

GROUND 1: <u>The rejection for claims 1, 2, and 4-27, based on 35 U.S.C. § 101</u>.

The Examiner rejects claims 1, 2, and 4-27 under 35 U.S.C. §101 as allegedly directed toward non-statutory subject matter as allegedly failing to provide a real world result.

Appellants respectfully point out that claim 1 describes a method that <u>increases the efficiency and/or running time</u> of a matrix subroutine, which is clearly a useful and practical result. It does so, by changing how the matrix is stored in the memory of the computer, and more particularly, by storing the matrix data in memory in units of blocks of data as based upon the cache working space. Moreover, unlike any prior art of record, the method of the claimed invention involves sizing the data block in memory to have one dimension larger than the larger dimension of the cache working space and the other dimension of the data block smaller than the smaller dimension of the working space.

On the bottom line on page 2 of the Office Action, the Examiner alleges: "Neither the data stored in the caches nor the result of executing matrix subroutine has a real world value."

Appellants respectfully submit that this statement is incorrect. The present invention is one of a series of inventions that are related to the problem of improving efficiency in the processing of matrix operations. How the matrix data gets into and out of the cache greatly effects how efficient the matrix subroutine will execute, once it is recognized that memory retrieval can be very inefficient when using standard format of computer languages such as Fortran and C.

It is implicit in claim 1 that the matrix is conventionally stored in one of the two standard formats of DLA. In the paper "High-performance linear algebra algorithms using new generalized data structures for matrices" by Fred Gustavson, it is shown that these standard data structures hurt the performance of matrix subroutines. These results are generally accepted by the Engineering and Scientific Community at large.

Broadly speaking, Claim 1 describes a better data structure for a matrix that the matrix subroutine will be processing. The result will be a more efficient execution of the matrix subroutine. Claim 1 is directed at changing the standard input data to the better matrix data structure of the invention.

The Examiner also seems to suggest that, in order for a claim involving matrix manipulation to pass muster as statutory subject matter, the claim must recite a real-world application being processed using that matrix processing.

Therefore, taking claim 1 as an example, the result of the claimed invention could be viewed either as the "increasing of efficiency/speed" or as the defined step for store data, as recited in the claim limitation. It is brought to the Examiner's attention that this increase in speed/efficiency can be physically measured and so, inherently has "real-world" application.

Therefore, again, Appellants respectfully submit that the claimed invention meets the criterion for statutory subject matter because its result is that of increasing computer efficiency, using the method described in the independent claims.

In view of the foregoing, the Board is respectfully requested to withdraw this rejection for non-statutory subject matter.

GROUND 2: <u>The rejection for claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27, based on 35 U.S.C. § 112, second paragraph</u>.

The Examiner alleges that claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27 are indefinite and rejected under 37 CFR §112, second paragraph, because the Examiner considers that the claim language "*... storing data contiguously ... in an increment block size ...*" is somehow unclear, since, the Examiner continues "*... data of a matrix contiguously stored in memory can be viewed as being stored in any increment size.*"

In response, Appellants respectfully point out that claim 1 states that the data structure of the present invention is contiguous in memory. Standard data structures for matrices store sub matrices in memory as non-contiguous data.

The concept of contiguous data storage as related to the blocks of the present invention is actually quite well described at various locations in the specification. For example, lines 6-8 of page 12 recite: "Each of the four NB/2 by 2NB sub-rectangles 301, which are stored row-wise, is preferably contiguous in memory and map well into the L1 cache."

Further, lines 14-18 on page 14 recite: "Although it might seem intuitive that a block size based on double the dimension of the L1 cache would hinder performance for matrix multiplication, the reason that this approach works is that the portion of the matrix brought into the cache with each block allows a complete multiplication of that block."

Lines 7-12 of page 15 recite: "Typically, in a machine in which memory is accessed in increments of line size, preferably the matrix data is laid out contiguously in memory in "stride one" form, where "stride one" means that the data is retrieved in increments of a line. Therefore, in machines having a line size, according to the present invention, the matrix data is preferably contiguously stored into and preferably retrieved from memory in increments of the machine line size."

Thus, the present invention involve data contiguously stored in memory so that an entire block of size NB/2 by 2NB will be substantially retrieved in a continuous and uninterrupted memory operation and some computers will actually transfer this block size of data as a single memory operation.

Therefore, Appellants respectfully submit that one having ordinary skill in the art would clearly understand the meaning of this claim terminology, particularly in view of the description in the specification.

Moreover, the Examiner's comment is not correct that "*... data of a matrix contiguously stored in memory can be viewed as being stored in any increment size.*"

Standard format of a matrix is a <u>one</u> dimensional layout. A matrix is a <u>two</u> dimensional object. Therefore, in this regard, one has the following theorem.

Theorem: Let A be an M by N matrix in standard format. It is impossible to store MB by NB submatrix B of A and its NB by MB transpose B' contiguously unless MB = LDA = M or MB = NB = 1.

That is, Lao does not mention LDA and, hence, by default sets LDA = M and NB = q where n*q = N. So, Lao chooses MB = M and NB = q for B. He then forms B transpose in-place via the use of permutations. In contrast, in the present invention, we choose both MB and NB independently of M and N and, hence, Lao is useless relative to the method of the present invention.

Moreover, an increment size less than memory line size will hurt performance. That is, since data is moved in increments of memory line size, typically 128 bytes, to be contiguous, data of interest must be stored in this increment. Thus, if there are 8 bytes of data of interest in a current retrieval request, then 8 bytes of the current retrieval will be useful and 120 bytes will be wasted. In contrast, the present invention includes the concept of placing data in memory in units of a block of data that is contiguous in its entirety for a next processing operation in the cache working space, as using block size dimensions that are clearly counterintuitive.

Thus, in contrast to matrix data as typically stored in memory, the present invention teaches to bring an entire block of data into cache, where the block size has been predetermined to fit into the cache working area even though the block has one dimension larger than the cache working area. There is no suggestion in the references of record to place matrix data in contiguous order in blocks of data to be moved into cache in units of these <u>blocks having the dimensions</u> described in the independent claims.

Therefore, Appellants respectfully submit that there is no lack of clarity in the claim wording when viewed in its entirety and respectfully request that the Board withdraw this rejection.

GROUND 3: <u>The rejection for claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27, based on 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,099,447 to Myszewski, further in view of US Patent Publication 2003/0088600 to Lao et al.</u>

The Examiner alleges that Myszewski, when modified by Lao, renders obvious the invention described by claims 1, 2, 5-10, 12-16, 18, 19, 21-23, and 25-27.

Appellants again respectfully disagree and submit that the rejection of record fails to meet the initial burden of a prima facie obviousness rejection, since, even if the two cited references were to be combined, the combination would still fail to satisfy the plain meaning of the claim language of even the independent claims.

That is, the present invention describes a general technique of dividing data into large data blocks related to the size of the working area of the cache. Moreover, as clearly described in the independent claims, the blocks of data of the present invention involve blocks having one dimension larger than the cache dimension and one dimension smaller than the cache dimension.

Neither reference cited by the Examiner suggests a block size with such dimensions, and the Examiner makes no attempt to point to any description to support the rejection. The description in paragraph [0071] of secondary reference Lao and the example therein does not satisfy the description of the block size described in the independent claims, since the partitions shown are related to the matrix size, not the cache size. Nor is there is no suggestion in paragraph [0071] to have a block dimension larger than the cache dimension.

Hence, turning to the clear language of the claims in neither Myszewski nor Lao is there any teaching or suggestion of: "...a <u>first dimension of said block being larger than a corresponding first dimension of said cache</u> and a <u>second dimension of said block being smaller than a corresponding second dimension of said cache</u>, such that <u>said block fits into a working space of said cache</u>", as required by independent claim 1. The remaining independent claims have similar language.

Moreover, Appellants respectfully submit that the Examiner does not understand the present invention. Both Myszewski and Lao are directed to matrices subroutines whose input matrices are in standard format. As Appellants have attempted to explain to the Examiner, it is impossible to partition a matrix in standard format in contiguous blocks unless one changes the layout of the standard format.

Neither Myszewski nor Lao does this in their patents.

As Applicants explained above, standard format of a matrix is a one dimensional layout. A matrix is a two dimensional object. That is, Lao does not mention LDA and, hence, by default sets LDA = M and NB = q where n*q = N. So, Lao chooses MB = M and NB = q for B. He then forms B transpose in-place via the use of permutations. In contrast, in the present invention, we choose both MB and NB independently of M and N and, hence, Lao is useless relative to the method of the present invention.

Moreover, an increment size less than memory line size will hurt performance. That is, since data is moved in increments of memory line size, typically 128 bytes, to be contiguous, data of interest must be stored in this increment. Thus, if there are 8 bytes of data of interest in a current retrieval request, then 8 bytes of the current retrieval will be useful and 120 bytes will be wasted. In contrast, the present invention includes the concept of placing data in memory in units of a block of data that is contiguous in its entirety for the next processing cycle in the cache working space, as using a block size that clearly is counterintuitive.

Thus, in contrast to matrix data as typically stored in memory, the present invention teaches to bring an entire block of data into cache, where the block size has been predetermined to fit into the cache working area even though the block has one dimension larger than the cache working area. There is no suggestion in the references of record to place matrix data in contiguous order in blocks of data to be moved into cache in units of these blocks having the dimensions described in the independent claims.

Regarding LAPACK matrix routines, the present invention is claiming a way to make the LAPACK matrix routine more efficient as LAPACK requires its input matrices to be in standard format. Current LAPACK are improved by Myszewski. The present invention improves LAPACK routines even further in a non obvious manner.

Regarding Claim 23, the present invention is about sub-matrices that are contiguous. The current art of matrix subroutines expects its matrices to be in standard format.

Therefore, Appellants submit that there are elements of the claimed invention that are neither taught nor suggested in the prior art of record, and the Examiner is respectfully requested to reconsider and withdraw this rejection.

Additionally, Appellants submit again the following comments relative to this prior art rejection, particularly since the prior art rejection does not address all claims.

The present invention teaches the technique of composite blocking, as articulated in the independent claims. The "double blocking" defined in claims 4, 11, 17, and 20 is a specific embodiment of composite blocking. There is no suggestion in Myszewski to convert and store in memory the matrix data of a matrix in square blocks larger than cache and consisting of rectangular blocks of contiguous data that will each fit into the working area of a cache (even though one dimension of the rectangular blocks is larger that its corresponding cache dimension).

To consider this prior art reference in a different perspective, a rectangular block makes DGEMM run faster. However, the DLAFA requires square blocking. The composite blocking of the present invention serves both purposes. Myszewski does not even recognize this disparate requirement and does not suggest a technique that satisfies this dual requirement, nor does it address DLAFA.

The technique of the present invention reduces the data thrashing that typically occurs when matrix data is simply retrieved in its original format as is required by higher level languages such as Fortran and C.

The data thrashing occurs, in the conventional processing, because portions of the matrix data are typically flushed from cache during processing so that additional memory accesses are required as the flushed data becomes needed for current processing and because matrix data is typically not contiguous as actually stored in memory. The present invention addresses this data thrashing by actually storing in memory the matrix data as rectangular contiguous sub blocks of a size that fits into cache and, if the matrix data is also contiguous, it will also be organized on memory line size, thereby increasing the speed at which future matrix data will enter the cache to be consumed in the matrix processing. This occurs because the data actually brought into cache from memory will be completely contiguous matrix data (rather than standard layout data in portions of some lines of data) and because the matrix subroutine processing will fit the processing result back into the data currently being consumed, rather than inadvertently flushing out matrix data not yet being processed.

In contrast to the conventional method of dealing with matrix data, the present invention teaches storing the matrix data in memory as actually being rectangular sub blocks of data that fits the sub blocks together so that all the matrix data is contiguous, where "contiguous" means that the data is in a good or optimal order to be used in its processing. The standard format of higher level conventional computer languages makes no attempt to place matrix data into its appropriate "contiguous" format.

Finally, relative to the Examiner's Response in paragraph 6 beginning at the top of page 5 of the Office Action, Appellants respectfully submit that, regarding subparagraph 2, the Examiner's statement is true. However, it is irrelevant, as the sub matrices of standard format are not contiguous in memory.

Regarding subparagraph 3, Myszewski must, in DGEMM, process standard matrix format as the API for DGEMM requires this format. In contrast, the current claim 1 is directed at a non standard format whose use will improve the efficiency of a matrix subroutine like DGEMM even further. Based on this observation, the Examiner's remark appears true, but it is irrelevant relative to the present invention. A careful reading of both Myszewski and Lao will reveal that they only consider standard matrix data layout, some of whose sub matrices are not contiguous.

Therefore, Appellants submit that there are elements of the claimed invention that are not taught or suggested by Myszewski, even if modified by Lao, and the Board is respectfully requested to reverse this rejection.

IX. CONCLUSION

In view of the foregoing, Appellants submit that claims 1, 2, and 4-27, all the claims presently pending in the application, are clearly enabled and patentably distinct from the prior art of record and in condition for allowance. Thus, the Board is respectfully requested to remove all rejections of claims 1, 2, and 4-27.

Please charge any deficiencies and/or credit any overpayments necessary to enter this paper to Assignee's Deposit Account number 50-0510.

Respectfully submitted,

[illegible]

Dated: March 31, 2008

Frederick E. Cooperrider
Reg. No. 36,769

McGinn Property Law Group, PLLC
8231 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer Number: 21254

CLAIMS APPENDIX

The claims, as reflected upon entry of the Amendment Under 37 CFR §1.111 filed on August 21, 2007, are shown below:

1. (Rejected) A method of increasing at least one of efficiency and speed in executing a matrix subroutine on a computer, said method comprising:

storing data contiguously for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache and a second dimension of said block being smaller than a corresponding second dimension of said cache, such that said block fits into a working space of said cache.

2. (Rejected) The method of claim 1, further comprising:

retrieving said data from said memory in units of said increment block; and

executing at least one matrix subroutine using said data.

3. (Canceled)

4. (Rejected) The method of claim 1, wherein said cache comprises a cache having a cache size CS and said block increment size comprises a block of size 2NB by NB/2, wherein NB2 $= \alpha$ CS, $\alpha < 1$, so that said block occupies a sizeable portion of said cache.

5. (Rejected) The method of claim 1, wherein said cache comprises an L1 or L2 cache, said L1 or L2 cache comprising a cache closest to at least one of a Central Processing Unit (CPU) and a Floating-point Processing Unit (FPU) of a computer system associated with said computer memory.

6. (Rejected) The method of claim 1, wherein said matrix data is loaded contiguously in said memory in increments of a memory line size LS and data is retrievable from said memory in units of LS.

7. (Rejected) The method of claim 2, wherein said at least one matrix subroutine comprises a matrix operation.

8. (Rejected) The method of claim 2, wherein said at least one matrix subroutine comprises a subroutine from a LAPACK (Linear Algebra PACKage).

9. (Rejected) The method of claim 2, wherein said subroutine operates on an increment block of data as a result of a single call on this data.

10. (Rejected) An apparatus, comprising:

a processor for processing a matrix subroutine;

a cache associated with said processor; and

a memory, wherein said memory stores data for memory calls of said matrix subroutine as contiguous data in an increment block size that is based on a dimension of said cache and loads said blocks of data into said cache for said matrix subroutine processing, wherein a dimension of said increment block size is larger than any dimension of a working area of said cache used for processing said matrix subroutine.

11. (Rejected) The apparatus of claim 10, wherein said cache comprises a cache having a cache size CS, and said block increment size comprises a block of size 2NB by NB/2, wherein NB2 $= \alpha$ CS, $\alpha < 1$, so that said block occupies a sizeable portion of said cache.

12. (Rejected) The apparatus of claim 10, wherein said matrix subroutine comprises a matrix operation.

13. (Rejected) The apparatus of claim 10, wherein said matrix subroutine comprises a subroutine from a LAPACK (Linear Algebra PACKage).

14. (Rejected) The apparatus of claim 10, wherein a line size of said memory is LS and data is retrieved from said memory in units of LS, each said block of data being retrieved by usually an integral number of memory line retrievals.

15. (Rejected) A computer program product for use with a computer, said computer program product comprising a machine-readable medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus, said instructions including a method of storing data for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache and a second dimension of said block being smaller than a corresponding second dimension of said cache, such that said block fits into a working space of said cache.

16. (Rejected) The computer program product of claim 15, wherein said matrix subroutine comprises a subroutine from a LAPACK (Linear Algebra PACKage).

17. (Rejected) The computer program product of claim 15, wherein said cache comprises a cache having a cache size CS, and said block increment size comprises a block of size 2NB by NB/2, wherein $NB2 = \alpha$ CS, $\alpha < 1$, so that said block occupies a sizeable portion of said cache.

18. (Rejected) The computer program product of claim 15, wherein a line size of said memory is LS and data is retrieved from said memory in units of LS, each said block of data being retrieved by usually an integral number of memory line retrievals.

19. (Rejected) A method of solving a problem using linear algebra, said method comprising at least one of:

initiating a computerized method of performing one or more matrix subroutines, wherein said computerized method comprises storing data for a matrix subroutine call in a computer memory in an increment block size that is based on a cache size of said computer, a first dimension of said block being larger than a corresponding first dimension of said cache and a second dimension of said block being smaller than a corresponding second dimension of said cache, such that said block fits into a working space of said cache;

transmitting a report from said computerized method via at least one of an internet interconnection and a hard copy; and

receiving a report from said computerized method.

20. (Rejected) The method of claim 19, wherein said cache comprises a cache having a size CS, and said block increment size comprises a block of size 2NB by NB/2, wherein NB2 = α CS, $\alpha < 1$, so that said block occupies a sizeable portion of said cache.

21. (Rejected) A method of providing a service, said method comprising an execution of a matrix subroutine in accordance with the method of claim 1.

22. (Rejected) A method of providing a service, said method comprising at least one of:

solving of a problem using linear algebra in accordance with the method of claim 19; and

providing a consultation to solve a problem that utilizes said computerized method.

23. (Rejected) The method of claim 1, wherein a size of said cache is CS and a working size of said cache for said matrix data is α CS, $\alpha < 1$, wherein said matrix to be processed comprises submatrices, and wherein data of said submatrices are stored in memory as k rectangular blocks of contiguous data that will each fit into said cache working size, k being an integer, and each said rectangular block having total size α CS.

24. (Rejected) The method of claim 23, wherein said rectangular block of data stored in said memory comprises a rectangular block of size 2NB by NB/2 of contiguous matrix data.

25. (Rejected) The method of claim 23, further comprising:

processing said rectangular blocks of matrix data by calling a DGEMM (Double-precision Generalized Matrix Multiply) kernel a plurality of times, using each one of said rectangular blocks of contiguous data with each said DGEMM kernel call.

26. (Rejected) The method of claim 25, wherein data for all operands used in said DGEMM kernel comprise data stored as contiguous data in lines of said memory such that data for each said operand can be retrieved as contiguous data from said memory using usually an integral number of memory line retrievals respectively appropriate for each said operand.

27. (Rejected) The method of claim 23, said method further comprising:

for data of said matrix, preliminarily converting and storing in said memory said data of said matrix into a number of rectangular blocks of contiguous data that will each fit into said cache size approximately NB x NB, including, if necessary, adding padding data to fill up a block or a complete line of memory, said padding chosen to have no effect on a calculation result of said matrix subroutine call.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None